Zarządzanie bezpieczeństwem aplikacji WWW
Opracowanie
Styczeń 2008 r.

Rational. software

IBM

Zrozumieć bezpieczeństwo aplikacji WWW

Spis treści

Firmy w coraz większym stopniu uzależniają swój rozwój od coraz bardziej złożonych aplikacji WWW, co sprawia, że coraz trudniej zapewnić im właściwy poziom bezpieczeństwa. Większość przedsiębiorstw chroni swoje serwisy WWW za pomocą firewalli, warstwy SSL, zabezpieczeń sieciowych oraz zabezpieczeń na poziomie hostów. Jednakże ataki są najczęściej wymierzone w same aplikacje, a wymienione technologie nie zapewniają im wystarczającej ochrony.

W tym opracowaniu wyjaśniono, w jaki sposób można chronić przedsiębiorstwo, oraz przedstawiono metodę lepszego zabezpieczania jego aplikacji WWW.

Co sprawia, że aplikacje WWW są narażone na ataki?

W modelu odniesienia OSI (Open System Interconnection)[1] każdy komunikat przechodzi przez siedem warstw protokołu sieciowego. Najwyżej znajduje się warstwa aplikacji. Obejmuje ona protokół HTTP oraz inne protokoły służące do przesyłania komunikatów z treścią, w tym HTML, XML, SOAP i usługi Web Services.

W tym dokumencie skoncentrowaliśmy się na atakach wymierzonych w aplikacje, przeprowadzanych z wykorzystaniem protokołu HTTP, z którymi tradycyjne firewalle nie radzą sobie zbyt dobrze. Wielu hakerów wie, w jaki sposób zakamuflować swoje żądania HTTP na poziomie sieci, aby wyglądały na niegroźne mimo że zawierają dane, które mogą wyrządzić szkody w systemie. Ataki przeprowadzane z wykorzystaniem protokołu HTTP mogą umożliwić hakerowi uzyskanie nieograniczonego dostępu do baz danych, wykonywanie dowolnych komend systemu, a nawet zmodyfikowanie zawartości serwisu WWW.

Najważniejsze informacje

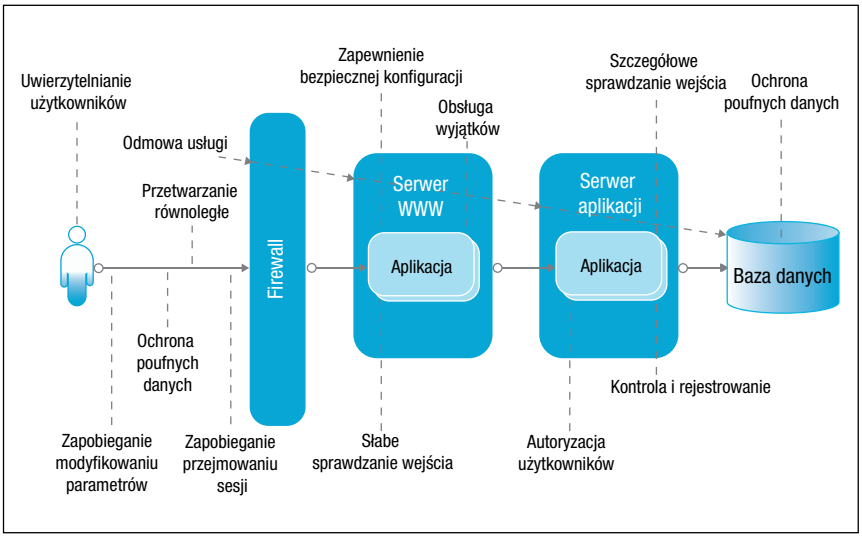
Bez odpowiednich środków, które pozwalają zarządzać testowaniem zabezpieczeń przez cały cykl tworzenia aplikacji, zespoły programistów mogą pozostawić w programach luki umożliwiające prowadzenie ataków z wykorzystaniem protokołu HTTP. Wynika to z następujących faktów:

- Analitycy i projektanci postrzegają bezpieczeństwo jako problem dotyczący sieci lub infrastruktury informatycznej, w wyniku, czego tylko kilku specjalistów ds. bezpieczeństwa zdaje sobie sprawę z istnienia zagrożeń na poziomie aplikacji.
- Zespoły formułują wymagania dotyczące bezpieczeństwa aplikacji w postaci niejasnych oczekiwań lub negacji (np. nie wolno dopuścić do pozostawienia niechronionych punktów wejścia). Powoduje to trudności przy opracowywaniu testów.
- Zabezpieczenia aplikacji są testowane na późnym etapie cyklu ich tworzenia, a do tego sprawdzana jest tylko ich ogólna odporność na włamania.

Przykłady typowych ataków wymierzonych w aplikacje WWW

Aby zabezpieczyć aplikacje WWW przed atakami, przedsiębiorstwa powinny stosować ogólne metody prewencyjne oraz przeznaczone specjalnie do tego celu technologie.

Rozwiązania wykorzystywane do ochrony aplikacji WWW powinny być wybierane na podstawie słabych punktów zabezpieczeń właściwych dla tych aplikacji. Na rysunku 1 przedstawiono wiele punktów w systemie, które mogą wymagać zabezpieczenia. Często najlepiej jest zastosować najpierw ogólne środki zapobiegawcze, aby upewnić się, że wybrano technologię najlepiej dostosowaną do faktycznych potrzeb, a nie tę, która podobno radzi sobie z najnowszymi technikami stosowanymi przez hakerów.



Rysunek 1. Zagadnienia związane z bezpieczeństwem aplikacji WWW

Najważniejsze informacje

Przedsiębiorstwa mogą zastosować wiele różnych środków zapobiegawczych, aby chronić aplikacje WWW przed naruszeniami bezpieczeństwa z wykorzystaniem technik podszywania się pod innego użytkownika, fałszowania danych lub zacierania śladów.

W tabeli 1 przedstawiono najczęściej spotykane zagrożenia oraz możliwe do zastosowania środki zapobiegawcze. Należy jednak pamiętać, że konkretne zagrożenia mogą się różnić w zależności od aplikacji.

Tabela 1. Najczęściej spotykane typy ataków wymierzonych w aplikacje WWW

Opis	Najczęstsze przyczyny	Środki zapobiegawcze
Podszywanie się		
Podanie danych uwierzytelniających innego użytkownika lub zmiana informacji cookie bądź parametru w celu podszycia się pod innego użytkownika lub oszukania systemu, że informacje cookie pochodzą z innego serwera.	• Stosowanie uwierzytelniania opartego na komunikacji przy udzielaniu dostępu do dowolnych danych użytkownika. • Brak szyfrowania danych uwierzytelniających, wskutek czego mogą one zostać przechwycone i wykorzystane przez osobę nieupoważnioną. • Przechowywanie danych uwierzytelniających w informacjach cookie bądź w postaci parametrów. • Stosowanie niesprawdzonych metod uwierzytelniania lub uwierzytelnianie z niewłaściwej zaufanej domeny. • Brak zgody na uwierzytelnienie hosta przez oprogramowanie klienckie.	Stosowanie rygorystycznego mechanizmu uwierzytelniania i ochrony danych uwierzytelniających z wykorzystaniem: • odpowiednich struktur udostępnianych przez systemy operacyjne; • szyfrowanych znaczników, takich jak informacje cookie sesji; • podpisów cyfrowych.
Fałszowanie danych		
Modyfikowanie lub usuwanie zasobów bez autoryzacji (np. zmiana wyglądu serwisu WWW lub modyfikowanie przesyłanych danych).	• Ufanie źródłom danych bez ich uprzedniego sprawdzenia. • Czyszczenie wejścia w celu zapobieżenia wykonaniu niepożądanego kodu. • Uruchamianie procesów ze zbyt wysokimi uprawnieniami • Brak szyfrowania danych poufnych.	• Wykorzystywanie zabezpieczeń systemu operacyjnego do blokowania dostępu do plików, katalogów i innych zasobów. • Sprawdzanie poprawności danych. • Mieszanie i podpisywanie przesyłanych danych (np. z wykorzystaniem warstwy SSL lub protokołu IPSec).
Zacieranie śladów		
Próby zniszczenia, ukrycia lub zmodyfikowania dowodów wykonania określonych działań (np. usuwanie dzienników, podszywanie się pod innego użytkownika w celu zażądania wprowadzenia zmian).	• Wykorzystywanie słabego mechanizmu autoryzacji i procesu uwierzytelniania lub niestosowanie ich w ogóle. • Nieprawidłowe rejestrowanie zdarzeń w dziennikach. • Umożliwianie przesyłania informacji poufnych za pomocą niezabezpieczonych kanałów komunikacyjnych.	• Stosowanie rygorystycznego mechanizmu uwierzytelniania, rejestrowanie transakcji w dziennikach i korzystanie z podpisów cyfrowych. • Kontrolowanie danych, dzienników itp.

Najważniejsze informacje

Środki zapobiegawcze mogą również posłużyć do obrony przed atakami, których celem jest uzyskanie dostępu do informacji poufnych i przeciążenie zasobów serwerowych.

Opis	Najczęstsze przyczyny	Środki zapobiegawcze
Ujawnianie informacji		
Ujawnianie informacji umożliwiających identyfikację osób (informacje PII), takich jak hasła czy dane kart kredytowych, oraz informacji dotyczących aplikacji źródłowej i/lub maszyn hosta.	• Pozwalanie uwierzytelnionemu użytkownikowi na uzyskiwanie dostępu do danych innych użytkowników. • Umożliwianie przesyłania informacji poufnych za pomocą niezabezpieczonych kanałów komunikacyjnych. • Stosowanie słabych algorytmów i kluczy szyfrowania.	• Przechowywanie informacji PII w ramach sesji (dane przejściowe) zamiast w postaci trwałej. • Stosowanie kodowania mieszającego oraz szyfrowania w odniesieniu do danych poufnych zawsze wtedy, gdy jest to możliwe. • Dopasowanie danych użytkownika do jego danych uwierzytelniających.
Odmowa usługi (DoS)		
• Zalewanie – wysyłanie dużej liczby komunikatów lub jednoczesnych żądań w celu przeciążenia serwera. • Blokowanie – wysyłanie ogromnej liczby żądań w celu wydłużenia czasu odpowiedzi serwera wskutek wykorzystania znacznej części zasobów albo w celu spowodowania restartu aplikacji.	• Umieszczanie na jednym serwerze zbyt wielu aplikacji lub aplikacji powodujących u siebie nawzajem konflikty. • Zaniedbywanie przeprowadzania obszernych testów jednostkowych.	• Filtrowanie pakietów za pomocą firewalla. • Używanie systemu równoważenia obciążenia w celu uzyskania kontroli nad liczbą żądań pochodzących z jednego źródła. • Używanie protokołów asynchronicznych do obsługi żądań o dużym zapotrzebowaniu na moc obliczeniową, a także mechanizmów naprawczych w przypadku błędów.
Uprawnienia wyższe od przysługujących		
Przekraczanie normalnych uprawnień dostępu w celu uzyskania praw administracyjnych lub dostępu do poufnych plików.	• Uruchamianie procesów serwera WWW jako użytkownik „root” lub „administrator”. • Wykorzystywanie błędów w kodzie w celu przepełnienia buforu i przejścia aplikacji w tryb debugowania.	• Korzystanie z możliwie jak najniższych uprawnień zawsze wtedy, gdy jest to możliwe. • Używanie bezpiecznych (type-safe) języków i opcji kompilatora w celu uniemożliwienia lub kontrolowania operacji przepełnienia buforu.

Najważniejsze informacje

Zastosowanie kilku prostych procedur może pomóc zespołom programistów zabezpieczyć aplikacje WWW przed najczęstszymi naruszeniami bezpieczeństwa i jednocześnie obniżyć koszty usuwania błędów tego rodzaju.

Podstawowe wytyczne dotyczące zabezpieczania aplikacji WWW

Zastosowanie przy tworzeniu aplikacji procesów właściwych dla bezpieczeństwa pozwala zespołom programistów chronić programy przed naruszeniami zabezpieczeń, takimi jak te omówione w tabeli 1.

W szczególności można zastosować kilka podstawowych wytycznych w odniesieniu do istniejących, nowych i modyfikowanych aplikacji w całym procesie, co z pewnością pomoże lepiej zabezpieczyć oprogramowanie i obniżyć koszty późniejszego usuwania błędów. Poniżej opisano pokrótce te wytyczne.

- ***Ustalenie i określenie podstawowych założeń:*** *przeprowadzenie kompletnego spisu aplikacji i systemów wraz z informacjami technicznymi (np. IP, DNS, używane systemy operacyjne) oraz biznesowymi (np. Kto dał zielone światło na wdrożenie aplikacji? Kto powinien zostać powiadomiony w przypadku awarii?). Następnie należy przeskanować całą infrastrukturę WWW w poszukiwaniu powszechnych słabych punktów zabezpieczeń i możliwości wykorzystania błędów w kodzie (exploit). Należy również sprawdzić serwery list i serwisy śledzące błędy programów w poszukiwaniu znanych ataków na używany system operacyjny, serwer WWW i pozostałe wykorzystywane produkty innych firm. Przed załadowaniem aplikacji na serwer należy się upewnić, że serwer został zabezpieczony i przeskanowany oraz że zainstalowano na nim najnowsze poprawki. Następnie należy przystąpić do skanowania aplikacji w poszukiwaniu słabych punktów zabezpieczeń pod względem znanych ataków i zwrócić przy tym szczególną uwagę na żądania HTTP oraz inne sposoby manipulowania danymi. Na koniec należy przetestować zawarte w aplikacji funkcje uwierzytelniania i zarządzania uprawnieniami użytkowników oraz wyłączyć wszystkie nieznane usługi.*
- ***Ocena i określenie ryzyka:*** *należy określić ryzyko związane z wykorzystaniem konkretnych aplikacji i systemów ze szczególnym uwzględnieniem składnic danych, kontroli dostępu, funkcji i obiektów udostępnianych użytkownikom oraz zarządzania uprawnieniami. Należy uszeregować wykryte słabe punkty zabezpieczeń aplikacji według ich priorytetu. Następnie należy przeanalizować zgodność aplikacji bądź systemu ze strategiami organizacyjnymi i branżowymi oraz regulacjami prawnymi. Ważne jest określenie operacji zarówno dopuszczalnych, jak i niedopuszczalnych.*
- ***Zabezpieczenie aplikacji i kontrolowanie szkód:*** *należy być zawsze na bieżąco ze znanymi zagrożeniami dla bezpieczeństwa i bezzwłocznie instalować wszelkie dostępne poprawki do aplikacji i elementów infrastruktury. Jeśli nie można usunąć problemu z bezpieczeństwem, należy zastosować firewall aplikacyjny, ograniczyć dostęp, wyłączyć aplikację lub przemieścić ją w celu zminimalizowania ryzyka.*
- ***Nieustanne monitorowanie i przeglądy:*** *należy określić harmonogram przeglądów w ramach udokumentowanego procesu zarządzania zmianami. Po zakończeniu jednego etapu analizy należy natychmiast rozpocząć następny.*

Najważniejsze informacje

Rational Unified Process oferuje wszechstronną, iteracyjną strukturę tworzenia aplikacji WWW, opartą na sprawdzonych procedurach branżowych.

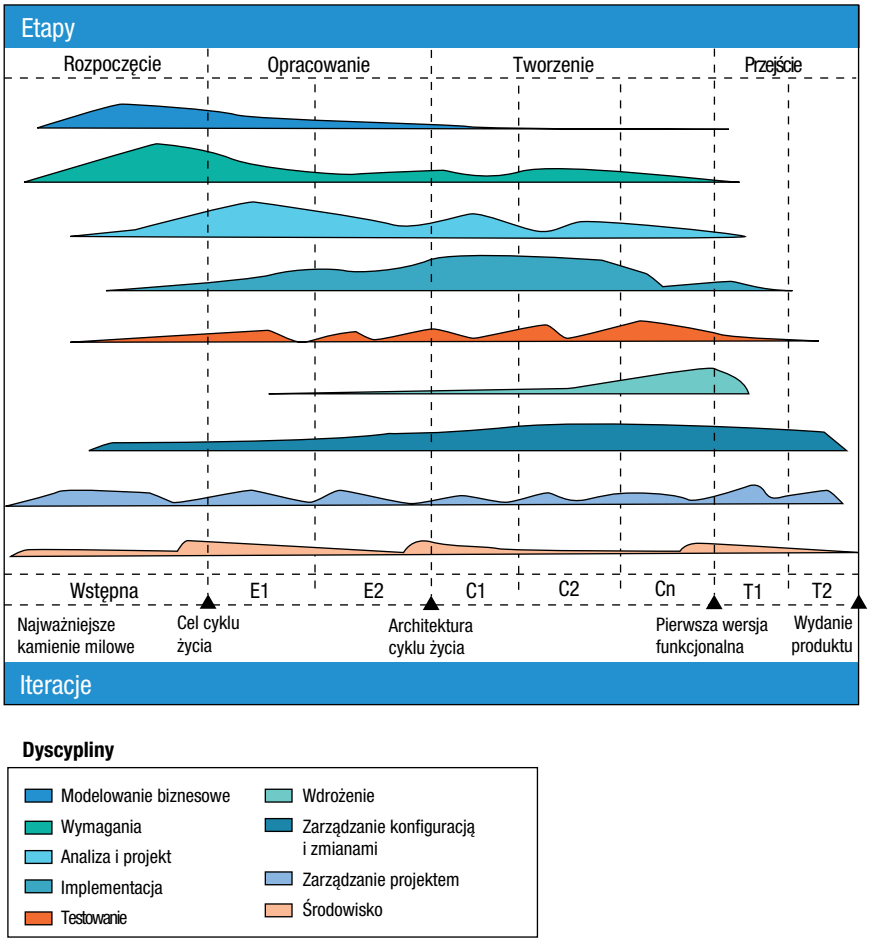
Zrozumieć cykl życia aplikacji WWW

Na rysunku 2 przedstawiono rozwiązanie IBM Rational® Unified Process® (IBM RUP®) – popularną, iteracyjną strukturę procesów tworzenia aplikacji WWW, opartą na sprawdzonych procedurach branżowych. W jej ramach przewidziano następujące podstawowe etapy (każdy z nich może wymagać do zakończenia dwóch lub większej liczby iteracji):

- ***Rozpoczęcie:*** *ustanowienie przykładu z praktyki biznesowej, zakresu i wizji operacyjnej. Następnie opracowanie początkowego modelu przypadków użycia, planu projektu, oceny ryzyka oraz opisu projektu, w tym podstawowych wymagań, wymagań w zakresie bezpieczeństwa (takich jak wyjaśnienie strategii bezpieczeństwa i zgodności z nimi), ograniczeń, cech oraz możliwych architektur prototypowych.*
- ***Opracowanie:*** *doprecyzowanie wizji, rozpatrzenie scenariuszy istotnych z punktu widzenia architektury w celu opracowania jej planu bazowego, jak również uszczegółowienie modelu przypadków użycia. Następnie utworzenie i przetestowanie jednego lub większej liczby prototypów w celu ograniczenia technicznych czynników ryzyka.*
- ***Tworzenie:*** *opracowanie szczegółowych projektów konkretnych składników i ich interakcji z innymi aplikacjami przy jednoczesnym, nieprzerwanym śledzeniu zgodności z wymaganiami. Następnie wygenerowanie kodu i przetestowanie składników pod kątem ich wydajności, niezawodności i bezpieczeństwa (przy jednoczesnym śledzeniu i rozwiązywaniu pojawiających się problemów), a na koniec zintegrowanie przetestowanych składników do pierwszej wersji rozwiązania.*
- ***Przejście:*** *wdrożenie aplikacji, przeszkolenie użytkowników i przeprowadzenie testów beta w celu zweryfikowania bezpieczeństwa i wydajności aplikacji oraz sprawdzenia jej zgodności z wymaganiami. Nieprzerwane monitorowanie wydajności, niezawodności i bezpieczeństwa w miarę wprowadzania kolejnych zmian w aplikacji.*

Najważniejsze informacje

Każdy z czterech etapów rozwiązania RUP (Rational Unified Process) — rozpoczęcie, opracowanie, tworzenie i przejście — obejmuje wiele dyscyplin i może wymagać więcej niż jednej iteracji.



Rysunek 2. Etapy, dyscypliny i kamienie milowe w IBM Rational Unified Process

Najważniejsze informacje

Przedsiębiorstwa korzystają z rozwiązania RUP w całym cyklu życia aplikacji WWW do proaktywnego wykrywania i usuwania słabych punktów zabezpieczeń, zanim staną się zbyt kosztowne do naprawienia.

W sytuacji, gdy presja ze strony rynku skłania organizacje do jak najszybszego tworzenia aplikacji WWW, bez należytego przetestowania ich zabezpieczeń, pozostawione słabe punkty zabezpieczeń mogą stanowić poważne zagrożenie dla przedsiębiorstwa.

Testowanie zabezpieczeń w całym cyklu życia aplikacji WWW

Zastosowanie się do wytycznych RUP już na wczesnym etapie tworzenia aplikacji WWW pozwala wykryć i usunąć słabe punkty zabezpieczeń odpowiednio wcześnie, gdy nie jest to jeszcze tak kosztowne.
W miarę postępu prac programistycznych i wdrożeniowych błędy w aplikacjach mogą się nawarstwiać, a ich usunięcie może być coraz bardziej kłopotliwe i drogie.

Przykładowo zespoły odpowiedzialne za zbieranie wymagań na etapach rozpoczęcia i opracowania mogą nie znać najczęstszych zagrożeń dla bezpieczeństwa aplikacji WWW, wskutek czego mogą zaniedbać wymagania, które mogłyby zapobiec ich wystąpieniu. Jeśli na etapie tworzenia nie będzie wyraźnie określonych wymagań w zakresie bezpieczeństwa, programiści mogą stosować nieoptymalny pod tym względem kod lub stworzyć nowy kod za pomocą kreatora zintegrowanego środowiska programistycznego (IDE), który nie uwzględnia w odpowiednim stopniu kwestii bezpieczeństwa. Następnie przy użyciu gotowego kodu mogą niewłaściwie sprawdzać poprawność danych lub źle interpretować opcje zabezpieczające w strukturze aplikacji. Na etapie przejścia organizacje często zlecają przeanalizowanie aplikacji WWW kilku specjalistom ds. bezpieczeństwa, którzy starają się wychwycić słabe punkty zabezpieczeń przed wdrożeniem aplikacji w środowisku produkcyjnym, przez co powstają wąskie gardła. Ponadto usunięcie wykrytych na tym etapie niedoskonałości i błędów jest z reguły bardzo kosztowne i pracochłonne.

Jak widać w tabeli 2[2], naprawienie błędu w projekcie po udostępnieniu aplikacji WWW jest około 30 razy droższe niż usunięcie tego samego błędu na etapie projektowania. Należy również zauważyć, że w przedstawionych wartościach szacunkowych w ogóle nie uwzględniono takich kosztów, jak utrata udziału w rynku, pogorszenie reputacji czy spadek poziomu zadowolenia klientów.

Najważniejsze informacje

Naprawienie błędu w projekcie po wdrożeniu aplikacji WWW jest około 30 razy droższe niż usunięcie tego samego błędu na etapie projektowania.

Aby uniknąć konieczności wprowadzania kosztownych poprawek, przedsiębiorstwa mogą zintegrować testowanie zabezpieczeń aplikacji ze swoimi procesami tworzenia i dostarczania oprogramowania.

Tabela 2. Względny koszt naprawy błędów w zależności od czasu ich wykrycia

Typ błędu	Projektowanie	Kodownie	Integracja	Testy beta	Wdrożenie
Błąd w projekcie	1x	5x	10x	15x	30x
Błąd kodowania		1x	10x	20x	30x
Problem z integracją			1x	10x	20x

Właściwe metody testowania

Aby uniknąć konieczności wprowadzania kosztownych poprawek, przedsiębiorstwa muszą zintegrować metody testowania zabezpieczeń aplikacji, takie jak proces przedstawiony na rysunku 3, ze swoimi procesami tworzenia i dostarczania oprogramowania (wraz z pozostałymi środkami do zarządzania jakością).

Tabela 3. Metody testowania zabezpieczeń aplikacji WWW

Opis	Zalety	Wady
Ręczne		
Testowanie penetracyjne lub na zasadzie zatwierdzania zabezpieczeń przez kilku specjalistów ds. zabezpieczeń z wykorzystaniem znanych narzędzi i skryptów.	Dobrze ukierunkowane testy konkretnych funkcji aplikacji.	• Testowaniem zajmują się wyłącznie specjaliści, co może powodować powstawanie wąskich gardeł. • Może prowadzić do przeoczenia wielu błędów i generowania wysokich, wielokrotnych kosztów w przyszłości. • Ze względu na ograniczenia czasowe testowane są tylko niektóre elementy aplikacji.
Zautomatyzowane		
Zwykle realizowane w jeden z następujących sposobów: • Wstępujące – konkretne testy pojedynczych funkcji tworzone przez programistę; • Zstępujące – testy opracowane przez zespoły ds. kontroli jakości z uwzględnieniem punktu widzenia użytkowników końcowych.	Rekompensata kosztów poprzez poprawę jakości, zmniejszenie nakładu pracy na testy odbiorcze oraz stosowanie iteracyjnych procesów tworzenia aplikacji.	Wymaga większych nakładów pracy na utworzenie i realizację niż w przypadku testowania ręcznego.

Najważniejsze informacje

W testach typu czarnej i białej skrzynki mogą być wykorzystywane narzędzia komercyjne, podczas gdy testy typu szarej skrzynki wymagają jednoznacznie zdefiniowanej struktury aplikacji.

Opis	Zalety	Wady
Testy typu czarnej skrzynki lub systemowe		
Sprawdzenie tylko danych na wejściu i wyjściu systemu oraz modyfikacja normalnego wejścia od użytkownika, aby wymusić na aplikacji niezamierzone przez programistów reakcje.	Korzystanie ze sprawdzonych, zautomatyzowanych narzędzi do przeprowadzania testów, które wymagają od osób je obsługujących minimalnej wiedzy o aplikacjach.	• Zastosowanie takiej metody jest możliwe tylko wtedy, gdy wszystkie elementy aplikacji są już gotowe do testowania (późny etap tworzenia oprogramowania lub środowisko produkcyjne). • Może generować transakcje trudne do zignorowania lub odtworzenia poprzez modyfikowanie danych wprowadzanych przez użytkowników. • Może przysłaniać błędy z powodu ograniczenia widoczności wewnętrznych mechanizmów funkcjonowania aplikacji.
Testy typu białej skrzynki lub źródłowe		
Analiza poszczególnych elementów w poszukiwaniu konkretnych błędów funkcjonalnych, często w połączeniu z narzędziami do skanowania kodu i równoległymi przeglądami.	Użycie narzędzi doskonale zintegrowanych ze środowiskami IDE wykorzystywanymi przez programistów, co pozwala właściwie sprecyzować wykrywanie błędów w testowanych funkcjach.	• Nie pozwalają wykryć błędów projektowych oraz niewłaściwie spełnionych lub niespełnionych wymagań. • Mogą nie wykryć słabych punktów zabezpieczeń przed atakami obejmującymi wiele składników lub przeprowadzanych w określonym czasie nieuwzględnionym w testach jednostkowych. • Zakłada się, że programiści wiedzą o testach zabezpieczeń, które należy przeprowadzić.
Testy typu szarej skrzynki (z wykorzystaniem struktury danej aplikacji)		
Połączenie metod stosowanych w testach typu białej i czarnej skrzynki w celu umożliwienia przeprowadzenia testów niedostępnych w narzędziach komercyjnych.	• Najbardziej wszechstronna metoda łącząca testy systemowe i jednostkowe. • Testy dotyczące konkretnych stanów i momentów, przeprowadzane z wykorzystaniem agentów lub elementów proxy. • Integracja struktury z aplikacją w celu monitorowania przepływu danych i kontrolowania ich bez wywierania wpływu na dane produkcyjne.	• Wymaga określenia struktury na etapie rozpoczęcia i podczas projektowania. • Opracowanie struktury testów może wymagać tyle samo pracy, co stworzenie samej aplikacji.

Najważniejsze informacje

Dzięki pomocy zewnętrznych konsultantów przedsiębiorstwa mogą prowadzić szkolenia oraz podejmować działania w zakresie wymiany informacji i monitorowania w celu propagowania wiedzy na temat bezpieczeństwa.

Cztery strategiczne sprawdzone procedury dotyczące ochrony aplikacji WWW

Aby rozwiązać problemy dotyczące bezpieczeństwa aplikacji WWW, organizacje mogą zastosować cztery obszerne, strategiczne sprawdzone procedury.

1. Propagowanie wiedzy na temat bezpieczeństwa

Obejmuje to szkolenia, wymianę informacji i monitorowanie, najlepiej we współpracy z konsultantem.

Szkolenia

Obejmuje to coroczne szkolenia z zakresu bezpieczeństwa dla wszystkich członków zespołu tworzącego aplikacje: programistów, specjalistów ds. kontroli jakości, analityków i menedżerów. W ramach szkoleń należy przedstawić najnowsze sposoby atakowania aplikacji, podać zalecany sposób zapobiegania im, a także omówić bieżące procedury dotyczące bezpieczeństwa w organizacji. Należy również wymagać od programistów udziału w szkoleniach, aby mogli opanować wstępnie zbudowane funkcje zabezpieczeń struktury. W przypadku szkoleń dotyczących gotowych produktów komercyjnych (ang. commercial off-the-shelf – COTS) należy korzystać z materiałów dostarczanych przez ich producentów. Szkolenia z zakresu bezpieczeństwa należy uwzględnić w planie projektu.

Najważniejsze informacje

Aby optymalnie wykorzystać ograniczone zasoby związane z bezpieczeństwem, przedsiębiorstwa mogą szeregować czynniki ryzyka i problemy z odpowiedzialnością według ich priorytetów.

Wymiana informacji
Należy zebrać sprawdzone procedury dotyczące bezpieczeństwa od wszystkich zespołów i ze wszystkich działów organizacji. Następnie należy je zebrać w skróconej formie i udostępnić w intranecie. Należy również zaangażować specjalistów ds. bezpieczeństwa informatycznego już na wczesnym etapie prac i opracować procesy obejmujące przekazywanie wiedzy współpracownikom. W każdym zespole tworzącym aplikacje powinna być jedna osoba z zespołu ds. bezpieczeństwa, która będzie pełnić funkcję osoby kontaktowej w sprawach wymagań i projektu aplikacji.

Monitorowanie
Należy się upewnić, że menedżerowie znają status bezpieczeństwa każdej tworzonej aplikacji. Aby wszystkie strony miały łatwy dostęp do aktualnych informacji, należy monitorować błędy w zabezpieczeniach przy użyciu firmowych infrastruktur śledzenia defektów i raportowania.

2. Sklasyfikowanie czynników ryzyka i odpowiedzialności
Każda organizacja dysponuje ograniczoną ilością zasobów i musi szeregować wszystkie sprawy według ich priorytetów. W ustalaniu priorytetów w przypadku bezpieczeństwa może pomóc:

- *zdefiniowanie wartości progowych czynników ryzyka i określenie sytuacji, w których zespół ds. bezpieczeństwa wyłączy usługi udostępniane przez aplikację;*
- *sklasyfikowanie aplikacji według czynników ryzyka (np. internetowa, intranetowa, ekstranetowa);*
- *generowanie okresowych raportów na temat czynników ryzyka na podstawie wyników skanowania zabezpieczeń, w których wykryte problemy będą przyporządkowywane do zdefiniowanych wartości progowych czynników ryzyka;*
- *utrzymywanie bazy danych, która pozwala analizować i klasyfikować aplikacje według czynników ryzyka, dzięki czemu można przekazywać zespołom informacje na temat wpływu ich aplikacji na bezpieczeństwo wdrożonych systemów.*

Najważniejsze informacje

Aby lepiej zarządzać procesami tworzenia i dostarczania aplikacji oraz zgodnością z przepisami, przedsiębiorstwa muszą wprowadzić programy związane z bezpieczeństwem oraz strategie ich egzekwowania bez tolerancji dla odstępstw.

Dzięki zintegrowaniu testów zabezpieczeń w całym cyklu dostarczania oprogramowania przedsiębiorstwa mogą usprawnić projektowanie, tworzenie i testowanie aplikacji.

3. Wprowadzenie strategii egzekwowania zasad z zerową tolerancją dla odstępstw

Nieodłącznym elementem zarządzania procesem tworzenia i dostarczania oprogramowania jest odpowiednio zdefiniowana strategia bezpieczeństwa. Strategia taka może zmniejszyć ryzyko wdrożenia aplikacji podatnej na ataki lub niezgodnej z przepisami. Na etapie rozpoczęcia należy ustalić, które testy aplikacja musi przejść przed wdrożeniem, i poinformować o tym wszystkich członków zespołu. Należy przeprowadzić formalny przegląd wymagań i specyfikacji projektowych pod kątem problemów z bezpieczeństwem jeszcze na etapach rozpoczęcia i opracowania – przed rozpoczęciem właściwych prac programistycznych. Wyjątki można dopuścić wyłącznie na etapie projektowania i wyłącznie po uzyskaniu odpowiedniego zatwierdzenia od kierownictwa.

4. Integracja testów zabezpieczeń z całym procesem tworzenia i dostarczania oprogramowania

Dzięki zintegrowaniu testów zabezpieczeń z całym cyklem dostarczania oprogramowania można osiągnąć zauważalny pozytywny wpływ na projektowanie, tworzenie i testowanie aplikacji. Wymagania funkcjonalne należy uzależnić od testów zabezpieczeń, jakie aplikacja musi przejść, przy jednoczesnym upewnieniu się, że struktura testów:

- *wykorzystuje zautomatyzowane narzędzia i umożliwia przeprowadzenie testów w dowolnym punkcie procesu tworzenia i dostarczania oprogramowania;*
- *uwzględnia testy jednostkowe, systemowe i aplikacyjne;*
- *umożliwia przeprowadzanie testów kontrolnych podczas tworzenia aplikacji;*
- *obejmuje testy sterowane zdarzeniami;*
- *wykorzystuje elastyczną metodykę programowania w odniesieniu do procedur związanych z bezpieczeństwem;*
- *może być stosowana na etapach tworzenia kodu, testowania i integrowania oraz w środowisku produkcyjnym.*

Najważniejsze informacje

Sprawdzanie poprawności kodu i danych może przynieść znaczne korzyści na etapach opracowania i tworzenia w procesie opracowywania oprogramowania.

W tabeli 4 przedstawiono sposoby opracowania (na etapie rozpoczęcia) wymagań uwzględniających szeroki zakres problemów z bezpieczeństwem aplikacji.

Tabela 4. Rozpoczęcie – zdefiniowanie wymagań w zakresie bezpieczeństwa

Problem	Kwestie do rozważenia podczas opracowywania ograniczeń i wymagań
Środowisko aplikacji	• Określenie, zrozumienie i zaadaptowanie firmowej strategii bezpieczeństwa. • Ustalenie ograniczeń dotyczących infrastruktury, takich jak usługi, protokoły i firewalle. • Ustalenie ograniczeń środowiska usług serwerowych (np. sieci VPN, środowiska testowe). • Określenie konfiguracji wdrożenia aplikacji. • Określenie struktur domeny sieciowej, grupowania zasobów i zdalnych serwerów aplikacji. • Zidentyfikowanie serwerów baz danych. • Ustalenie, jakie funkcje bezpiecznej komunikacji są obsługiwane w danym środowisku. • Rozważenie kwestii związanych z farmą serwerów WWW (w tym zarządzanie stanem sesji, klucze szyfrowania specyficzne dla danych maszyn, warstwa SSL, problemy z wdrażaniem certyfikatów, profile mobilne itp.); jeśli aplikacja wykorzystuje warstwę SSL, należy wskazać ośrodek certyfikacji (CA) oraz typy certyfikatów, jakie mają być wykorzystywane. • Określenie wymaganych kryteriów dotyczących skalowalności i wydajności. • Zbadanie poziomu zaufania do kodu.
Sprawdzanie poprawności wejścia i danych oraz uwierzytelnianie	• Założenie, że wszystkie dane wprowadzane z poziomu klienta są potencjalnie niebezpieczne. • Ustalenie granic zaufania do kont użytkowników oraz zasobów przekraczających te granice. • Określenie strategii zarządzania kontami oraz strategii kont o najmniejszych możliwych uprawnieniach. • Określenie wymagań dotyczących silnych haseł oraz środków egzekwowania tych wymagań. • Szyfrowanie danych uwierzytelniających użytkowników przy użyciu warstwy SSL, sieci VPN, protokołu IPSec itp.; zapewnienie, że informacje uwierzytelniające (np. znaczniki, informacje cookie, bilety) nie będą przesyłane z wykorzystaniem połączeń nieszyfrowanych. • Zapewnienie zwrotu do klienta minimalnej ilości informacji o błędzie w przypadku nieudanego uwierzytelnienia.
Zarządzanie sesjami	• Ograniczenie czasu ważności sesji. • Zabezpieczenie stanu sesji przed dostępem bez uprawnień. • Zapewnienie, że identyfikatory sesji nie są przekazywane w łańcuchach zapytań.

Najważniejsze informacje

Na etapie rozpoczęcia projektu, przedsiębiorstwa mogą opracować wymagania pozwalające usunąć wiele problemów z bezpieczeństwem aplikacji.

W tabeli 5 przedstawiono działania podejmowane na etapach opracowania i tworzenia, które są zgodne z określonymi wymaganiami w zakresie bezpieczeństwa.

Tabela 5. Opracowanie i tworzenie – modelowanie i programowanie zabezpiecze}

Zagadnienie	Etap opracowania	Etap tworzenia
Procedury dotyczące tworzenia kodu		• Nie należy obniżać ani zmieniać domyślnych ustawień zabezpieczeń bez przeprowadzenia odpowiednich testów w celu poznania wpływu zmian na bezpieczeństwo aplikacji. • Nie należy zakładać, że niejasność tajnych danych zapewni ich bezpieczeństwo; nie należy umieszczać takich informacji bezpośrednio w kodzie. • Nie należy prezentować informacji, które w danej chwili nie są niezbędne. • Należy często testować oprogramowanie pod kątem błędów w zabezpieczeniach i jak najszybciej usuwać wykryte problemy. • Wszelkie niepowodzenia wykonania operacji powinny być przekierowywane do trybu bezpiecznego; nie należy wyświetlać zawartości stosu ani pozostawiać danych poufnych bez ochrony.
Sprawdzanie poprawności wejścia i danych	• Wszystkie dane wprowadzane na poziomie klienta należy traktować jako podejrzane; trzeba koniecznie sprawdzać ich poprawność na serwerze kontrolowanym przez aplikację, nawet w przypadku wykonywania wstępnego sprawdzenia poprawności po stronie klienta. • Należy rozwiązać wszystkie potencjalne problemy z iniekcją kodu SQL oraz osadzaniem kodu (XSS). • Należy określić punkty wejścia oraz granice zaufania.	• Należy sprawdzać poprawność wszystkich parametrów wejściowych, w tym parametrów pochodzących z pól, łańcuchów zapytań, informacji cookie i nagłówków HTTP. • Należy akceptować wyłącznie znane dobre dane wejściowe i odrzucać znane złe dane wejściowe. • Należy sprawdzać poprawność typu, długości, formatu i zasięgu danych. • Należy upewnić się, że wyjście zawierające dane wejściowe jest poprawnie zakodowane (HTML lub adres URL).

Najważniejsze informacje

Ulepszenie procedur dotyczących uwierzytelniania, autoryzacji i zarządzania konfiguracją pomaga przedsiębiorstwom rozwiązywać problemy z bezpieczeństwem już na etapach opracowania i tworzenia.

Zagadnienie	Etap opracowania	Etap tworzenia
Uwierzytelnianie	• Należy oddzielić dostęp do obszarów publicznych i o ograniczonym dostępie, kont z przypisaną tożsamością oraz zasobów przekraczających zdefiniowane granice zaufania. • Należy wyznaczyć konta osób serwisujących aplikację lub administrujących nią. • Należy zapewnić szyfrowanie i bezpieczne przechowywanie danych uwierzytelniających użytkowników. • Należy określić identyfikator do celów uwierzytelniania w bazie danych.	• Należy przechowywać hasła w postaci skróconej. • Należy zwracać tylko konieczną minimalną ilość informacji o błędach uwierzytelniania. • Nie należy wykorzystywać informacji zawartych w nagłówkach HTTP dla celów bezpieczeństwa.
Autoryzacja	• Należy określić wszystkie identyfikatory kont i zasoby, do których poszczególne konta mają dostęp. • Należy wskazać uprzywilejowane zasoby i uprzywilejowane operacje. • Należy oddzielić uprawnienia przypisane różnym rolom (np. wbudowana granulacja autoryzacji). • Należy określić wymagania w zakresie bezpieczeństwa dotyczące dostępu do kodu.	• Należy ograniczyć logowanie do bazy danych do procedur składowanych związanych z dostępem do danych; nie należy pozwalać na uzyskanie bezpośredniego dostępu do tabel. • Należy ograniczyć dostęp do zasobów na poziomie systemu.
Zarządzanie konfiguracją	• Należy zabezpieczyć interfejsy oraz zdalne kanały administracyjne za pomocą silnego mechanizmu uwierzytelniania i funkcji autoryzacji. • Należy stosować uprawnienia administratorów oparte na rolach. • Kontom związanym z procesami i usługami należy nadawać możliwie jak najmniejsze uprawnienia.	• Należy zabezpieczać składnice danych konfiguracyjnych. • Nie należy przechowywać danych poufnych w niezaszyfrowanych tekstowych plikach konfiguracyjnych.

Najważniejsze informacje

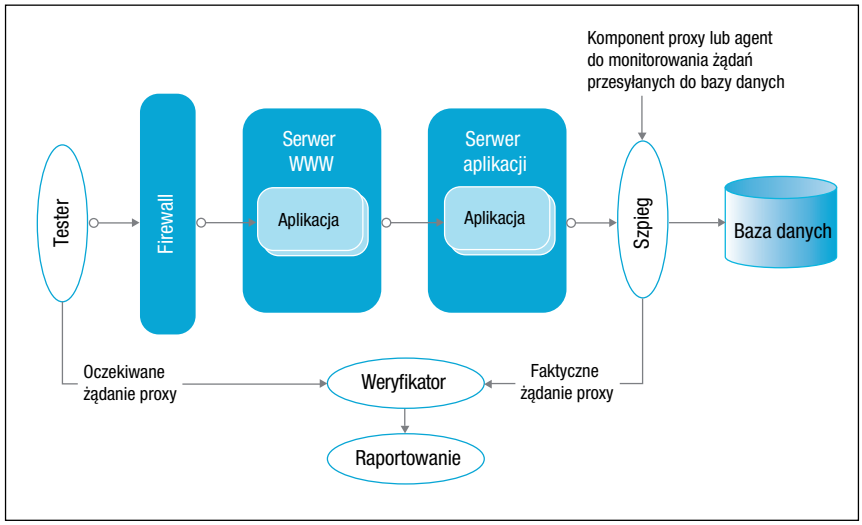
Zabezpieczanie sesji, zarządzanie wyjątkami oraz kontrola i rejestrowanie danych w dziennikach również mogą ułatwić zwiększenie bezpieczeństwa aplikacji WWW.

Zagadnienie	Etap opracowania	Etap tworzenia
Zabezpieczenie danych poufnych oraz sesji	• Należy unikać przechowywania tajnych danych; w odniesieniu do wszystkich danych, które muszą zostać zapisane, należy określić algorytmy szyfrowania i wielkości kluczy. • Należy określić mechanizmy zabezpieczające dane poufne przesyłane przez sieć. • Należy używać warstwy SSL do zabezpieczania informacji cookie dotyczących uwierzytelniania oraz do szyfrowania ich zawartości. • Należy określić metodykę wspomagającą zabezpieczanie kluczy szyfrowania i zapewniającą używanie wyłącznie znanych bibliotek i usług kryptograficznych. • Należy określić odpowiednie algorytmy szyfrowania oraz wielkości kluczy.	• Nie należy przechowywać danych poufnych w kodzie aplikacji. • Nie należy przechowywać połączeń do bazy danych, haseł ani kluczy w postaci jawnego tekstu. • Nie należy rejestrować danych poufnych w postaci jawnego tekstu. • Nie należy przechowywać danych poufnych w informacjach cookie ani przekazywać ich w postaci łańcucha zapytania lub pola formularza.
Zarządzanie wyjątkami	• Należy zdefiniować standardową metodę ustrukturyzowanej obsługi wyjątków. • Należy określić ogólne komunikaty o błędzie, które mają być zwracane klientowi.	• Po wystąpieniu wyjątku należy ujawniać możliwie jak najmniej informacji.
Kontrola i rejestrowanie	• Należy określić kluczowe parametry kontroli i rejestrowania. • Należy określić funkcje przechowywania, bezpieczeństwa i analizy w odniesieniu do plików dzienników. • Należy określić sposób przepływu tożsamości użytkownika wywołującego przez kolejne warstwy – na poziomie systemu operacyjnego lub aplikacji.	• Nie należy rejestrować danych poufnych.

Najważniejsze informacje

Testowanie sterowane zdarzeniami może umożliwić przedsiębiorstwom zintegrowanie testów zabezpieczeń z tworzeniem aplikacji.

Oprócz uczynienia bezpieczeństwa integralną częścią procesu tworzenia i dostarczania aplikacji można zintegrować testy zabezpieczeń z tworzoną aplikacją, aby móc przeprowadzać testy sterowane zdarzeniami. W tym przypadku, gdy użytkownik przesyła żądanie, a aplikacja na nie odpowiada, test porównuje odpowiedź z odpowiedzią oczekiwaną lub wcześniej zapisaną, aby ustalić, czy system działa prawidłowo. Na rysunku 3 przedstawiono taką przykładową sytuację: aplikacja wykorzystuje bazę danych jako komponent zaplecza. Tester umieszcza w przepływie żądania szpiegowski komponent proxy oraz weryfikator, któremu „mówi", jak powinno wyglądać normalne żądanie, tak, aby mógł on porównać je z żądaniem wychwyconym przez komponent proxy.



Rysunek 3. Testowanie zabezpieczeń aplikacji WWW wykorzystującej wewnętrzną bazę danych

Dowolna usługa – poczta elektroniczna, XML lub wcześniejsza usługa – może posłużyć jako zaplecze. Sposób zaimplementowania kodu do przeglądania żądań zależy od architektury aplikacji. Komponent szpiegujący może na przykład być fałszywym obiektem dostępu do danych, komponentem proxy lub klasą dziedziczącą z usługi frontowej. Można również utworzyć kod specjalnie na potrzeby testu i wstawić go w strumień danych w celu zapewnienia danych raportowania wymaganych przez środowisko testowe. Koordynacja obiektów testujących zapewnia pełną, precyzyjną kontrolę nad wieloma różnymi testami. Testy te można przeprowadzać jako testy typu czarnej lub białej skrzynki, co zwiększa szanse wykrycia problemów z bezpieczeństwem na wczesnym etapie tworzenia i dostarczania aplikacji, zanim zamienią się w poważne zagrożenie dla przedsiębiorstwa.

Więcej informacji

Więcej informacji o metodyce IBM Rational oraz możliwych sposobach tworzenia bezpiecznych aplikacji WWW z wykorzystaniem zautomatyzowanych narzędzi bezpieczeństwa IBM Rational można uzyskać u przedstawiciela IBM lub pod adresem: **ibm.com/software/pl/rational**

IBM Polska Sp. z o.o.
ul. 1 Sierpnia 8
02–134 Warszawa
tel. (+ 48 22) 878 67 77
faks (+ 48 22) 878 68 88

Strona główna IBM znajduje się pod adresem: **ibm.com/pl**

Wyprodukowano w Polsce

[1] Międzynarodowa Organizacja Normalizacyjna; www.iso.org.
[2] www.nist.gov/director/prog-ofc/report02-3.pdf.

RAW14010-USEN-00